individual jobs in a parameter study experiment. The NASA X-38 Crew Return Vehicle (upper right) was the subject of a two-dimensional parameter study in Mach number and angle of attack. ILab generated and submitted 192 separate flow-field computations for the requested 16 values of Mach number and 12 values of angle of attack. Pictured (bottom right) is the surface of coefficients of lift-over-drag for the X-38 at these 192 parameter combinations.

Point of Contact: Maurice Yarrow
(650) 604-5708
yarrow@nas.nasa.gov

Multithreading for Dynamic Unstructured Grid Applications

Rupak Biswas

The success of parallel computing in solving realistic computational applications relies on their efficient mapping and execution on large-scale multiprocessor architectures. When the algorithms and data structures corresponding to these problems are unstructured or dynamic in nature, efficient implementation on parallel machines offers considerable challenges. Unstructured applications are characterized by irregular data-access patterns whereas dynamic mesh adaptation causes computational work-loads to grow or shrink at run time. For such applications, dynamic load balancing is required in order to achieve algorithmic scaling on parallel machines. Our objectives were to implement various parallel versions of a dynamic unstructured algorithm and to critically compare their performances in terms of run time, scalability, programmability, portability, and memory overhead.

A multithreaded version of a dynamic unstructured mesh-adaptation algorithm has been implemented on the Cray (formerly Tera) Multithreaded Architecture (MTA). Multithreaded machines can tolerate memory latency and utilize substantially more of their computing power by processing several threads of computation. For example, the MTA processors each have hardware support for up to 128 threads, and are therefore especially well suited for irregular and dynamic applications. Unlike traditional parallel machines, the MTA has a large uniform shared memory, no data cache, and is insensitive to data placement. Parallel programmability is significantly simplified since users have a global view of the memory, and need not be concerned with partitioning and load-balancing issues. Performance was compared with an MPI implementation on the T3E and the Origin2000, and a shared-memory directives-based implementation on the Origin2000.

A standard computational mesh simulating flow over an airfoil was used for our experiments to compare the three parallel architectures. The initial mesh, consisting of more than 28,000 triangles, was refined a total of five times to generate a mesh 45 times larger, as shown in figure 1. Performance by platform and programming paradigm is presented in figure 2. It is important to note that different parallel versions use different dynamic load-balancing strategies. The multithreaded implementation of the adaptation algorithm required adding a trivial amount of code to the original serial version and had little memory overhead. In contrast, the MPI version doubled the size of the code and required significant additional memory for the communication buffers. The simulation on the eight-processor MTA at San Diego Supercomputing Center using 100 threads per processor was almost ten times faster than that obtained on 160 processors of a T3E, and more than 100 times faster than that on a 64-processor Origin2000 running a directives-based version. Results indicate that multithreaded systems offer tremendous potential for quickly and efficiently solving some of the most challenging real-life problems on parallel computers.

Point of Contact: Rupak Biswas
(650) 604-4441
rbiswas@nas.nasa.gov

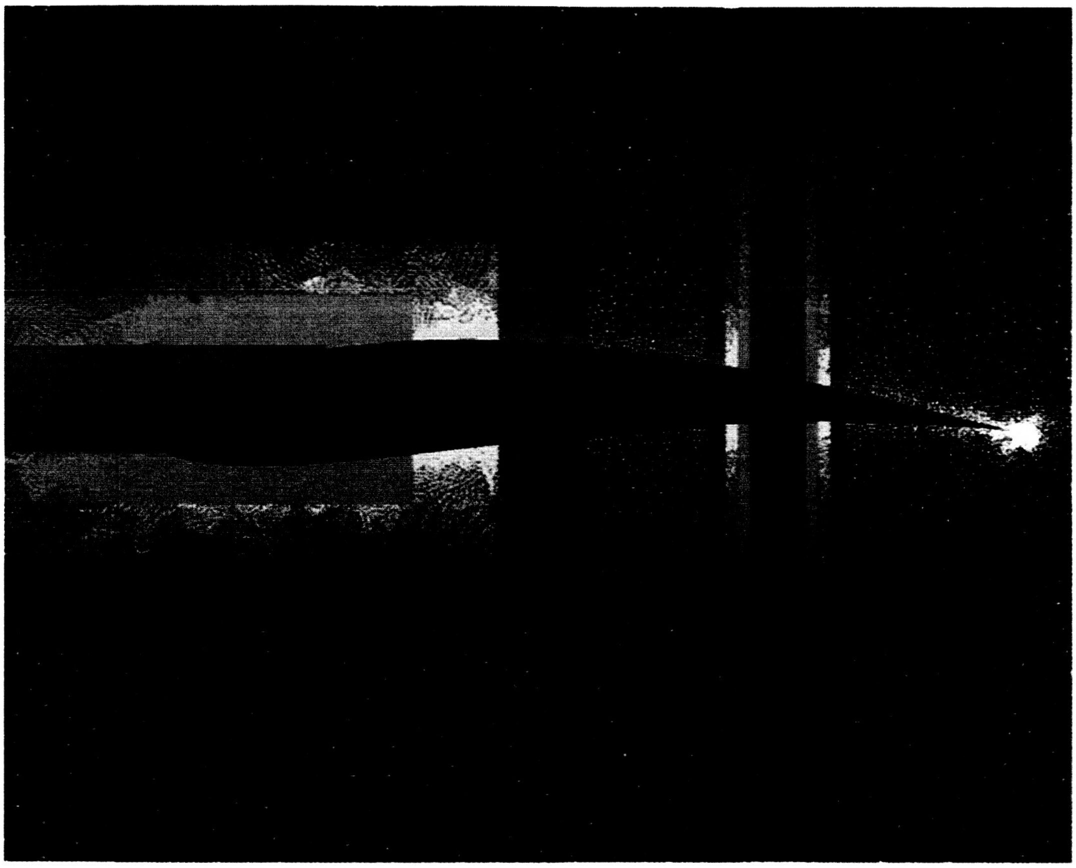
Fig. 1. A close-up view of the computational mesh that was geometrically refined in specific regions to better capture fine-scale phenomena.

Paradigm	System	Code increase	Memory increase	Parallel Performance					
MPI Message-passing	T3E	100%	70%	P	Refine	Overhead	Total		
				8	4.53	14.44	18.97		
				160	0.61	2.39	**3.00**		
				512	0.14	4.95	5.09		
	Origin 2000	100%	70%	P	Refine	Overhead	Total		
				8	8.31	11.62	19.93		
				64	1.41	3.99	**5.40**		
Shared-memory directives	Origin 2000	10%	5%	P	Refine	Overhead	Total		
				1	20.8	21.1	41.9		
				8	17.0	22.6	**39.6**		
				64	42.9	29.6	72.5		
Multi-threading directives	MTA	2%	7%		Threads per processor				
				P	1	40	60	80	100
				1	150	3.82	2.72	2.22	2.04
				2		1.98	1.40	1.15	1.06
				4		1.01	0.74	0.64	0.59
				8		0.55	0.41	0.37	**0.35**

Fig. 2. Performance of the dynamic unstructured mesh-adaptation algorithm by platform and programming paradigm. All times are in seconds; the fastest times are highlighted.

Overset Grid-Generation Software Package

William Chan, Stuart Rogers

In computer simulations of flows about an object, a computational grid is used to model the object's geometry. The Chimera overset-grid method is currently one of the most computationally cost-effective options for obtaining accurate simulations of flow involving complex geometry configurations, viscous fluid dynamics, and bodies in relative, dynamic motion. Considerable success has been achieved in applying this method to a wide variety of problems. The objective of the current work has been to develop a comprehensive set of software tools for performing pre- and post-processing of overset grids for complex-geometry simulation problems, for both static and dynamic cases. These tools have been packaged together in the Chimera Grid Tools software.

The Chimera Grid Tools package allows a user to create overset computational grids, and to perform geometry processing, grid diagnostics, solution analysis, and flow-solver input preparation. This package has been requested by and distributed to over 200 U.S. organizations under nondisclosure agreements, and has been utilized in aerospace, marine, automotive, environmental, and sports applications. The software consists of a hierarchy of modules together with documentation and examples. At the lowest level are libraries containing commonly used functions such as input/output routines for data files, stretching functions, projection routines, and many others. At one level above the libraries are about 30 independent programs that can be used in batch mode. Capabilities offered by these programs include editing, redistribution, smoothing and projection of grids, hyperbolic and algebraic surface and volume grid generation, and Cartesian grid generation. At the highest level is a graphical user interface called OVERGRID.